DOCUMENT RESUME

ED 057 560 EM 009 397

| | |
|---|---|
| TITLE | Project Solo; Newsletter Number Sixteen. |
| INSTITUTION | Pittsburgh Univ., Pa. Dept. of Computer Science. |
| SPONS AGENCY | National Science Foundation, Washington, D.C. |
| PUB DATE | 17 May 71 |
| NOTE | 7p.; See also ED 053 566 |
| EDRS PRICE | MF-$0.65 HC-$3.29 |
| DESCRIPTORS | *Computer Assisted Instruction; *Computer Programs; Programing; *Secondary School Students; *Set Theory; *Student Developed Materials |
| IDENTIFIERS | *Project Solo |

ABSTRACT

A computer program is presented that was written by an eighth grade student. It is capable of "writing" 16 bar melodies. A Project Solo module on set theory is also presented which is a tutorial computer-assisted approach to elementary set theory. (JY)

# PROJECT SOLO

AN EXPERIMENT IN REGIONAL COMPUTING FOR SECONDARY SCHOOL SYSTEMS



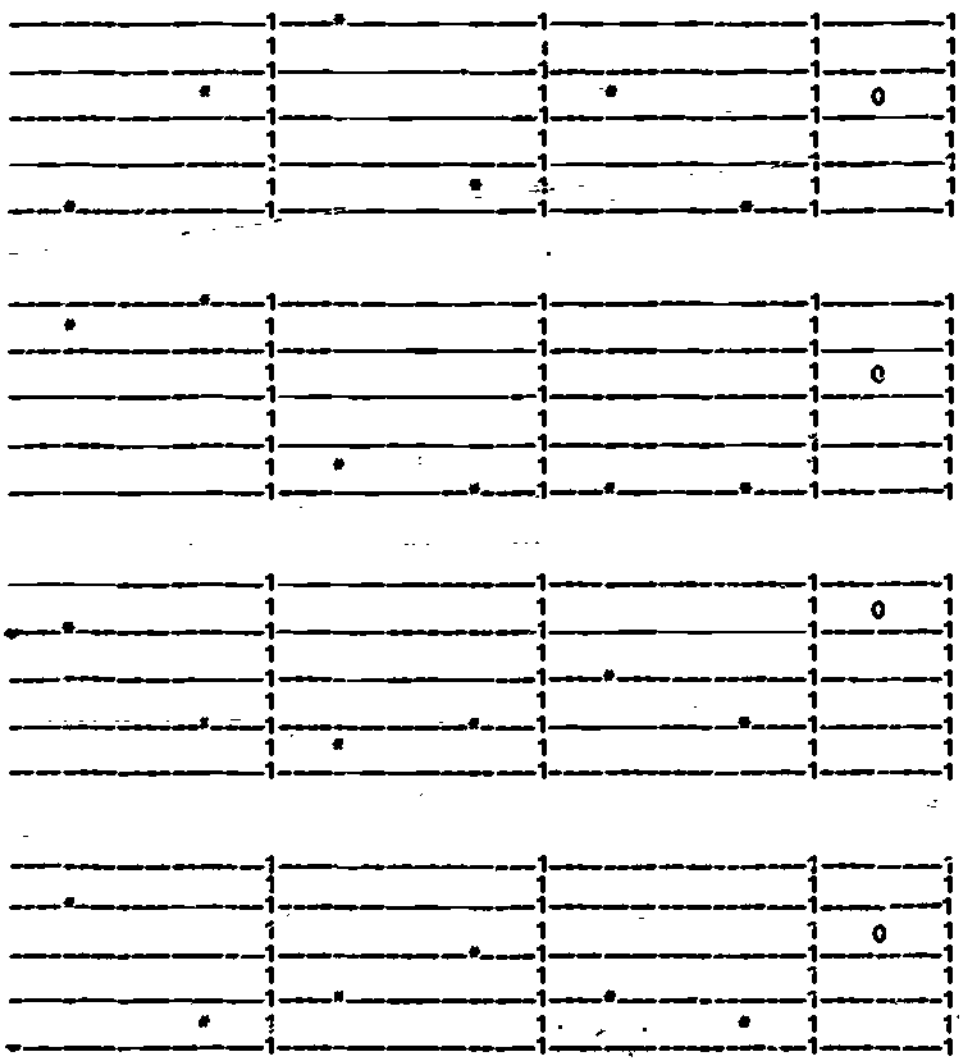
University of Pittsburgh • Department of Computer Science • Pittsburgh, Pennsylvania 15213

Newsletter No. 16 May 17, 1971

## Computing and the Renaissance Man

The program which "writes" 16 bar melodies of the kind shown below was written by Rob Croop (Grade 8). It is a nice example of the use of string arrays. It also illustrates the broad range of application possible with educational computing. A new melody is produced on every run, since the random generator NUM selects the notes. Mathematics teachers might want to have students modify the program so that the notes are selected as elements in various mathematical sequences. The notes can be kept within the range of the staff by letting, X=MOD(X-1, 9)+1 (cf. NEWBASIC Primer, p. 6-6). [See example in box on next page.]

THE BALLAD OF PROJECT SOLO



A listing of 166DS /MUSIC/

```
3 PRINT "        THE    BALLAD  OF  PROJECT.  SOLO"
4 PRINT
5 FOR R=1 TO 4
7 LET P=5
10 DIM A$(70,9)
20 LET B$="-"
22 LET  Q$=" "
30 FOR L=1 TO 70
35 GOSUB 415
36 NEXT L
37 LET Q$="1" LET B$=Q$
40 FOR L=20 TO 60 STEP 20
45 GOSUB 415
46 NEXT L
47 FOR L=70 TO 70
48 GOSUB 415
49 NEXT L
50 LET X=NUM(9)
55 LET V$="*"
56 IF P=65 LET V$="O"
57 LET A$(P,X)=V$
58 LET P=P+10
59 IF P>65 GOTO 70 ELSE 50
70 FOR S=9 TO 1 STEP -1
75 FOR T=1 TO 70
80 PRINT A$(T,S):
85 NEXT T
90 PRINT
95 NEXT S
96 PR. PR. PR.
100 NEXT R
105 END
```

**Modification to use the harmonic series to generate notes:**

```
1   N=0
50 N=N+1 LET X=1/N LET X=
INT(MOD(100*X-1, 9)+1)
```

```
415 LET A$(L, 9)=B$
420 LET A$(L, 8)=Q$
425 LET A$(L, 7)=B$
430 LET A$(L, 6)=Q$
435 LET A$(L, 5)=B$
440 LET A$(L, 4)=Q$
445 LET A$(L, 3)=B$
450 LET A$(L, 2)=Q$
455 LET A$(L, 1)=B$
460 RETURN
```

## A New Module on Set Theory (by Frank Wimberly)

An extremely powerful and expensive computer is currently being used in a large metropolitan school system to drill elementary school students in basic arithmetic. While this use of a computer may have motivational advantages over workbook or classroom drill and practice, it seems disappointing that a device which is capable of tens of thousands of decisions and calculations per second is used in an application which requires only a few such decisions and calculations per minute.

The work described here represents one of the ways in which we are using computers in a tutorial or simulation mode so as to take better advantage of their speed and power. The module called "Set-Theoretic Expressions" enclosed with this newsletter is based on such a tutorial.

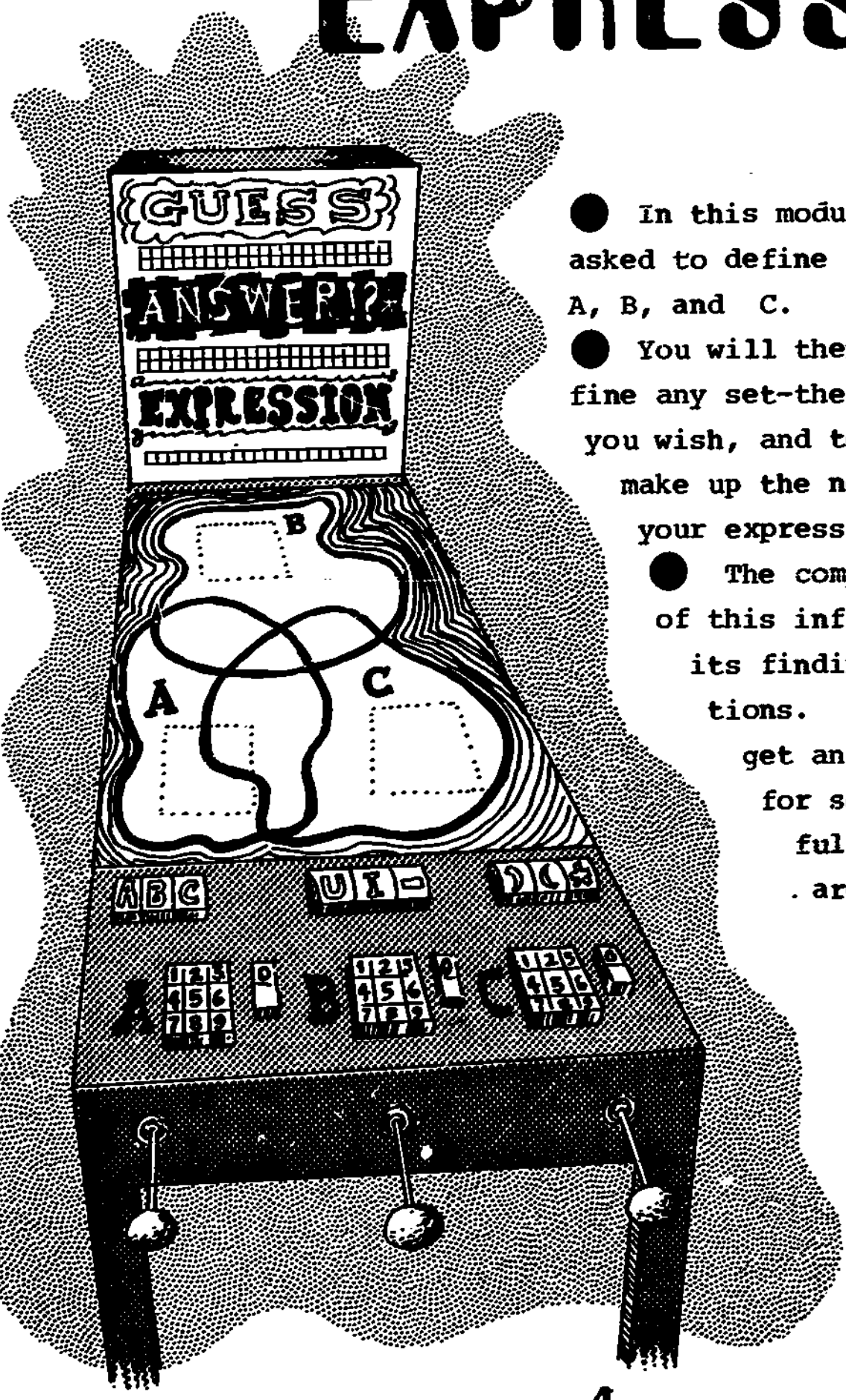
It is designed to encourage beginning students to explore elementary set theory. The module explains the use of the program and suggests some problems which can be solved with the help of the program. A listing of the tutorial (written in NEWBASIC) is also available. Since it is obviously a long and complicated program, our intention is to eventually store the binary code on the system, but only after feedback from users. For the present, run the program from NBS as shown.

To write a program in such generality that it accepts a fairly large class of expressions requires the use of parsing algorithms similar to the ones used in computer language compilers for interpreting arithmetic expressions. In any interaction the computer is kept busy doing what it does best, namely computing. The ratio of central processor units to connect time for this program is about ten times as great as it is for typical tutorial programs.

The freedom of the student to specify the expressions as well as the elements of the sets extends his power in such a way that he can explore set theory effectively in two ways. When he is asked what he thinks the answer is he can either 1) work through the expression and give an answer or 2) he can say that he thinks the answer is the empty set. In this second case the computer will list all the elements of the set represented by the expression and say that he left them out. As an example of how this latter use can be instructive, consider one of the exercises suggested in the written material. A series of equalities of set theoretic expressions is listed together with the statement that some of them are laws of set theory and some are not. It should be a challenge to define the sets in such a way that an equality is false, or failing that to understand why it is true. The point is the student can experiment, using the computer to avoid the tedious work that this experimentation would otherwise involve.

The Project Solo philosophy does of course suggest that students gain deeper insights when they write their own programs. Another module that will be based on the work described here is one that will assign the task of writing a program which will evaluate some fixed expression, with only the elements of the sets being variable. The core of this program will be three subroutines which perform the three operations. Note that it is impossible to accomplish this without having achieved a thorough understanding of the concepts union, intersection, and relative complement. It would probably involve too much work which is out of the mainstream to expect students in a mathematics course to write the more general program. However, if a secondary level course in computer science is developed (as we envision) the program could be the basis for a series of interesting and important modules concerned with "parsing algorithms". Such algorithms are behind the ability that a high level language like NEWBASIC has to "understand" programs written by people, and translate them into machine instructions.

# SET-THEORETIC EXPRESSIONS




- In this module you will first be asked to define three sets called A, B, and C.
- You will then be invited to define any set-theoretic expression you wish, and to say which elements make up the new set defined by your expression.
- The computer will digest all of this information, and compare its findings with your predictions. In this way you can get an experimental feel for some of the more useful relationships that are found in set theory.

*PROJECT SOLO*
*Department of*
*Computer Science*
*University of*
*Pittsburgh (15213)*
*Module 0103*

## EXPLORING SET THEORY

This program makes it possible to perform experiments with sets by allowing you to specify sets of integers and operations to be performed on them. Refer to the sample below to find out how to run the program and how to define a set. First log in, using the instructions given you by your teacher, then type the following (not underlined):

```
-NBS
VER. MAR 24 12:47
>RUN 166FW /PARSE2/
DEFINE THE SET A
?14,27,13,64,999 (carriage return)
```

The set A has been defined to be {14,27,13,64}. Note that 999 is not an element of A. You type 999 to tell the program that the set A has all the elements that you want it to have. If you wanted A to be the empty set you would type 999 right after the question mark and then hit carriage return. After you have defined the set A you will define two other sets, B and C, in exactly the same way. When you have finished defining the sets A, B, and C, the computer will next type:

```
WHAT SET THEORETIC EXPRESSION INVOLVING A,B, AND
C WOULD YOU LIKE TO COMPUTE?
?
```

You must then type an expression here according to the following rules:

1. "U" means union, "I" means intersection, and "-" means relative complement.
2. For each operation (U, I, or -) in the expression, there must be a pair of parentheses surrounding the two operands (A, B, or C). Examples:
   (A U B)*  ((A U B) I (A-C))*
   ((A U B)I C)*  (((A U B) I C)-(A U C))*
3. The expression must end with "*".
4. Spaces are ignored, so you may use them as needed for readability.

To specify the expression "A union B, intersect C," you would type:

? ((A U B)I C)*

As soon as the computer reads the expression it will ask:

WHAT DO YOU THINK THE ELEMENTS OF ((A U B)I C)* ARE?

?

You should list what you think the elements are, just as you did in defining the sets A, B, and C. Type 999 after the last element.

Complete Sample Run (underlines left off)

```
>RUN

DEFINE THE SET A
?1,2,3,4,5,999
NOW DEFINE B
?4,5,6,7,8,999
NOW DEFINE C
?7,8,9,1,2,999

WHAT SET THEORETIC EXPRESSION INVOLVING A, B, AND
C WOULD YOU LIKE TO COMPUTE?
?(A U B)*
WHAT DO YOU THINK THE ELEMENTS OF (A U B)* ARE?
?1,2,3,4,5,6,7,8,999
VERY GOOD. THAT'S EXACTLY RIGHT.
WOULD YOU LIKE TO COMPUTE ANOTHER EXPRESSION?
?YES

WHAT SET THEORETIC EXPRESSION INVOLVING A, B, AND
C WOULD YOU LIKE TO COMPUTE?
?((A - B)-C)*
WHAT DO YOU THINK THE ELEMENTS OF ((A - B)-C)* ARE?
?3,6,9,999
YOU INCLUDED THE FOLLOWING WHICH ARE NOT IN ((A - B)-C)*
 6     9
WOULD YOU LIKE TO COMPUTE ANOTHER EXPRESSION?
?NO
>LOGOUT
```

Problem 1

If A = {2,4,6,8,12,14,16,18,20}

B = {4,8,12,16,20}

C = {0,5,10,15,20}

Compute the following expressions:

((A I B)U C)*

((A U C)I B)*

(A - (B I C))*

((A I C) U (B I C))*

((A - B) U (A - C))*

Problem 2

Consider the following sets: A--high performance cars B--economy cars, C--cars with air-cooled engines.

| A | B | C |
|---|---|---|
| Corvette | Volkswagen | Volkswagen |
| Road Runner | Renault | Porsche 914 |
| Porsche 914 | Vega | Fiat 850 |
| BMW | BMW | Corvair |
| Triumph Spitfire | Porsche 914 | |
| Camaro Z-28 | Triumph Spitfire | |
| Shelby Mustang | Fiat 850 | |
| Firebird Formula 400 | | |
| Fiat 850 | | |

If someone asked you for a list of high-performance, economy cars without air-cooled engines, could you use the program to help him? (HINT: assign numbers to the brands in such a way that no two have the same number.)

Problem 3

Some of the following are laws of set theory and some are not. Try to find the false ones by defining A, B, and C in such a way that they aren't true.

((A U B)U C) = (A U(B U C))  ((A I B)UC) = ((A I C)U(B I C))

((A I B)U C) = (A I(B U C))  (A-(B U C)) = ((A-B) I (A-C))

(A-(B U C)) = ((A-B) U (A-C))

Note: If you are using the computer simply to evaluate expressions rather than to compare your answers with its answers, you can type 999 right away and claim that the set represented by the expression is empty. The computer will then list the elements and say that you left them out.